Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 04-036

Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization..

Navaratnasothie Selvakkumaran and George Karypis

September 29, 2004

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
29 SEP 2004		**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization..	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Army Research Laboratory,2800 Powder Mill Road,Adelphi,MD,20783-1197**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**25**	

Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization *

Navaratnasothie Selvakkumaran George Karypis

Abstract

In this paper we present a family of multi-objective hypergraph partitioning algorithms based on the multilevel paradigm, which are capable of producing solutions in which both the cut and the maximum subdomain degree are simultaneously minimized. This type of partitionings are critical for existing and emerging applications in VLSI CAD as they allow to both minimize and evenly distribute the interconnects across the physical devices. Our experimental evaluation on the ISPD98 benchmark show that our algorithms produce solutions that when compared against those produced by hMETIS have a maximum subdomain degree that is reduced by up to 36% while achieving comparable quality in terms of cut.

1 Introduction

Hypergraph partitioning is an important problem with extensive applications to many areas, including VLSI design [5], efficient storage of large databases on disks [32], information retrieval [37], and data mining [12, 20]. The problem is to partition the vertices of a hypergraph into k equal-size subdomains, such that the number of the hyperedges connecting vertices in different subdomains (called the *cut*) is minimized. The importance of the problem has attracted a considerable amount of research interest and over the last thirty years a variety of heuristic algorithms have been developed that offer different cost-quality trade-offs. The survey by Alpert and Kahng [5] provides a detailed description and comparison of various such schemes. Recently a new class of hypergraph partitioning algorithms has been developed [11, 14, 2, 19, 4], that are based upon the multilevel paradigm. In these algorithms, a sequence of successively smaller hypergraphs is constructed. A partitioning of the smallest hypergraph is computed. This partitioning is then successively projected to the next level finer hypergraph, and at each level an iterative refinement algorithm (*e.g.*, KL [26] or FM [13]) is used to further improve its quality. Experiments presented in [2, 19, 4, 35, 3, 9, 25] have shown that multilevel hypergraph partitioning algorithms can produce substantially better solutions than those produced by non-multilevel schemes.

However, despite the success of multilevel algorithms in producing partitionings in which the cut is minimized, this cut is not uniformly distributed across the different subdomains. That is, the number of hyperedges that are being cut by a particular subdomain (referred to as the *subdomain degree*) is significantly higher than that cut by other subdomains.

*This work was supported in part by NSF CCR-9972519, EIA-9986042, ACI-9982274, ACI-0133464, and ACI-0312828; the Digital Technology Center at the University of Minnesota; and by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under Cooperative Agreement number DAAD19-01-2-0014. The content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

	4-way	8-way	16-way	32-way	64-way
ibm01	1.27	1.55	1.60	1.70	1.76
ibm02	1.35	1.35	1.43	1.51	1.55
ibm03	1.18	1.43	1.68	1.70	1.84
ibm04	1.28	1.35	1.41	1.72	2.39
ibm05	1.16	1.17	1.24	1.33	1.41
ibm06	1.22	1.46	1.46	1.50	1.63
ibm07	1.29	1.46	1.79	1.94	2.04
ibm08	1.06	1.22	1.45	1.73	2.12
ibm09	1.09	1.23	1.65	1.91	2.31
ibm10	1.23	1.43	1.69	1.78	1.85
ibm11	1.21	1.55	1.54	1.66	2.02
ibm12	1.26	1.47	1.72	2.10	2.15
ibm13	1.31	1.81	1.66	1.91	1.85
ibm14	1.20	1.47	1.46	1.63	1.96
ibm15	1.28	1.51	1.71	1.87	2.09
ibm16	1.22	1.39	1.45	1.70	1.84
ibm17	1.18	1.42	1.52	1.80	2.13
ibm18	1.16	1.61	2.33	2.65	2.78

Table 1: The ratios of the maximum subdomain degree over the average subdomain degree of various solutions for the ISPD98 benchmark.

This is illustrated in Table 1 that shows the ratios of the maximum subdomain degree over the average subdomain degree of various k-way partitionings obtained for the ISPD98 benchmark [3] using the state-of-the-art hMeTiS [22] multilevel hypergraph partitioning algorithm. In many cases, the resulting partitionings contain subdomains whose degree is up to two times higher than the average degree of the remaining subdomains.

For many existing and emerging applications in VLSI CAD, producing partitioning solutions that both minimize the cut and also minimize the maximum subdomain degree is of great importance. For example, within the context of partitioning-driven placement, since the maximum subdomain degree of a partition (also known as the *bin degree*) can be considered a lower bound on the number of routing resources that is required, being able to compute partitions that both minimize the number of interconnects (which is achieved by minimizing the cut) and also evenly distribute these interconnects across the physical device to eliminate high density interconnect regions (which is achieved by minimizing the maximum subdomain degree) can significantly reduce the peak demand of routing resources and thus, help in reducing the peak congestion [17, 38]. Similarly, in the context of multi-chip configurations, a partitioned design cannot be mapped on to a set of chips, if there is a partition that exceeds the number of available I/O pins. For example, in the multi-chip configuration shown in Figure 1, if the degree of a subdomain exceeds 20, it cannot be mapped to any of the available chips. To address this problem, techniques based on pin-multiplexing have been developed [6] that allow multiple signals to go through the same I/O pins by using time division multiplexing. However, this approach reduces the speed at which the system can operate (due to time division) and increases the overall system design (due to the extra logic required for the multiplexing). However, the costs associated with pin-multiplexing can be significantly reduced and even eliminated by computing a decomposition that significantly reduces the maximum number of I/O pins required by any given partition.

In this paper we present a family of hypergraph partitioning algorithms based on the multilevel paradigm that are capable of producing solutions in which both the cut and the maximum subdomain degree are simultaneously minimized. Our algorithms treat the minimization of the maximum subdomain degree as a multi-objective optimiza-

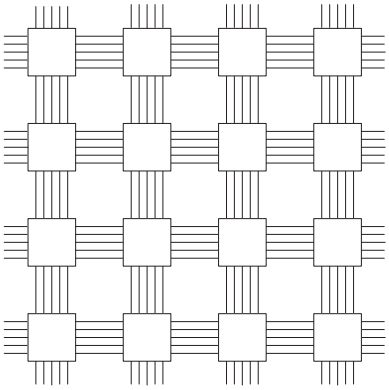

Figure 1: An example of multi-chip systems with grid interconnect topology

tion problem that is solved once a high-quality, cut based, k-way partitioning has been obtained. Toward this goal, we present highly effective multi-objective refinement algorithms that are capable to produce solutions that explicitly minimize the maximum subdomain degree and ensure that the cut does not significantly increase. This approach has a number of inherent advantages. First, by building upon a cut-based k-way partitioning, it leverages the huge body of existing research on this topic, and it can benefit from future improvements. Second, because the initial k-way solution is of extremely high-quality, it allows the algorithm to focus on minimizing the maximum subdomain degree without being overly concerned about the cut of the final solution. Finally, it provides a user-adjustable and predictable framework in which the user can specify how much (if any) deterioration on the cut he or she is willing to tolerate in order to reduce the maximum subdomain degree. We experimentally evaluated the performance of these algorithms on the ISPD98 [3] benchmark and compared them against the solutions produced by hMETiS [22]. Our experimental results show that our algorithms are capable of producing solutions whose maximum subdomain degree is lower by 5% to 36% while producing comparable solutions in terms of cut. Moreover, the computational complexity of these algorithms is relatively low, requiring on the average no more than twice the amount of time required by hMETiS in most cases.

The rest of the paper is organized as follows. Section 2 provides some definitions, describes the notation that is used throughout the paper, and provides a brief description of the multilevel graph partitioning paradigm. Section 3 discusses the various issues arising with minimizing the maximum subdomain degree and formally defines the two multi-objective formulations used in this paper. Sections 4 and 5 describe the *direct* and *aggressive* multi-phase refinement algorithms that we developed to simultaneously minimize the maximum subdomain degree and the cut of the resulting partitioning. Section 6 experimentally evaluates these algorithms and compares them against hMETiS. Finally, Section 7 provides some concluding remarks and outlines directions of future research.

2 Definitions and Notation

A *hypergraph* $G = (V, E)$ is a set of vertices V and a set of hyperedges E. Each hyperedge is a subset of the set of vertices V. The *size* of a hyperedge is the cardinality of this subset. A vertex v is said to be *incident* on a hyperedge e, if $v \in e$. Each vertex v and hyperedge e has a weight associated with them and they are denoted by $w(v)$ and $w(e)$, respectively.

A decomposition of V into k disjoint subsets $V_1, V_2, \ldots, V_k$, such that $\bigcup_i V_i = V$ is called a k-*way partitioning*

of V. We will use the terms *subdomain* or *partition* to refer to each one of these k sets. A k-way partitioning of V is denoted by a vector P such that $P[i]$ indicates the partition number that vertex i belongs to. We say that a k-way partitioning of V *satisfies a balancing constraint* specified by $[l, u]$, where $l < u$, if for each subdomain V_i, $l \leq \sum_{v \in V_i} w(v) \leq u$. The *cut* of a k-way partitioning of V is equal to the sum of the weights of the hyperedges that contain vertices from different subdomains. The *subdomain degree* of V_i is equal to the sum of the weights of the hyperedges that contain at least one vertex in V_i and one vertex in $V - V_i$. The *maximum subdomain degree* of a k-way partitioning is the highest subdomain degree over all k partitions. The *sum-of-external-degrees* (abbreviated as SOED) of a k-way partitioning is equal to the sum of the subdomain degrees of all the partitions. A net is said to be *exposed* w.r.t. a subdomain when it contributes to its degree.

Given a k-way partitioning of V and a vertex $v \in V$ that belongs to partition V_i, its *internal degree* denoted by $\text{ID}_i(v)$ is equal to the sum of the weights of its incident hyperedges that contain only vertices from V_i, and its *external degree* with respect to partition V_j denoted by $\text{ED}_j(v)$ is equal to the sum of the weights of its incident hyperedges whose all remaining vertices belong to partition V_j.

The *k-way hypergraph partitioning* problem is defined as follows. Given a hypergraph $G = (V, E)$ and a balancing constraint specified by $[l, u]$, compute a k-way partitioning of V such that it satisfies the balancing constraint and minimizes the cut. The requirement that the size of each partition satisfies the balancing constraint is referred to as the *partitioning constraint*, and the requirement that a certain function is optimized is referred to as the *partitioning objective*.

2.1 The Multilevel Paradigm for Hypergraph Partitioning

The key idea behind the multilevel approach for hypergraph partitioning is fairly simple and straightforward. Multilevel partitioning algorithms, instead of trying to compute the partitioning directly in the original hypergraph, first obtain a sequence of successive approximations of the original hypergraph. Each one of these approximations represents a problem whose size is smaller than the size of the original hypergraph. This process continues until a level of approximation is reached in which the hypergraph contains only a few tens of vertices. At this point, these algorithms compute a partitioning of that hypergraph. Since the size of this hypergraph is quite small, even simple algorithms such as Kernighan-Lin (KL) [26] or Fiduccia-Mattheyses (FM) [13] lead to reasonably good solutions. The final step of these algorithms is to take the partitioning computed at the smallest hypergraph and use it to derive a partitioning of the original hypergraph. This is usually done by propagating the solution through the successive better approximations of the hypergraph and using simple approaches to further refine the solution.

In the multilevel partitioning terminology, the above process is described in terms of three phases. The *coarsening phase*, in which the sequence of successively approximate hypergraphs (*coarser*) is obtained, the *initial partitioning phase*, in which the smallest hypergraph is partitioned, and the *uncoarsening and refinement phase*, in which the solution of the smallest hypergraph is *projected* to the next level finer graph, and at each level an iterative refinement algorithm such as KL or FM is used to further improve the quality of the partitioning. The various phases of multilevel approach in the context of hypergraph bisection are illustrated in Figure 2.

This paradigm was independently studied by Bui and Jones [8] in the context of computing fill-reducing matrix reordering, by Hendrickson and Leland [15] in the context of finite element mesh-partitioning, and by Hauck and Borriello [14] (called Optimized KLFM), and by Cong and Smith [11] for hypergraph partitioning. Karypis and Kumar extensively studied this paradigm in [23, 21, 24] for the partitioning of graphs. They presented novel graph coarsening schemes and they showed both experimentally and analytically that even a good bisection of the coarsest graph alone

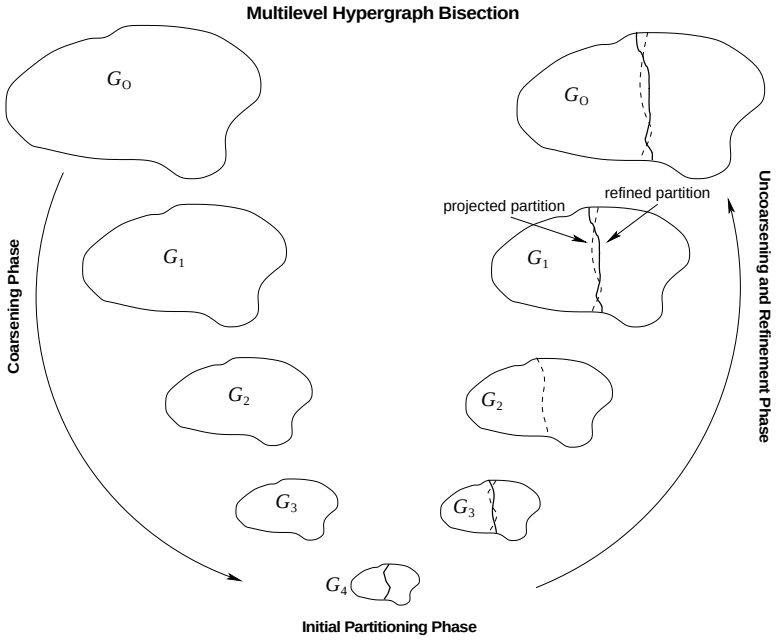

Figure 2: The various phases of the multilevel hypergraph bisection. During the coarsening phase, the size of the hypergraph is successively decreased; during the initial partitioning phase, a bisection of the smaller hypergraph is computed; and during the uncoarsening and refinement phase, the bisection is successively refined as it is projected to the larger hypergraphs. During the uncoarsening and refinement phase, the dashed lines indicate projected partitionings and dark solid lines indicate partitionings that were produced after refinement.

is already a very good bisection of the original graph. These coarsening schemes made the overall multilevel paradigm very robust and made it possible to use simplified variants of KL or FM refinement schemes during the uncoarsening phase, which significantly speeded up the refinement process without compromising overall quality. MeTiS [23], a multilevel graph partitioning algorithm based upon this work, routinely finds substantially better partitionings than other popular techniques such as spectral-based partitioning algorithms [28, 7], in a fraction of the time required by them. Karypis *et al* [19] extended their multilevel graph partitioning work to hypergraph partitioning. The hMeTiS [22] package contains many of these algorithms and have been shown to produce high-quality partitionings for a wide-range of circuits.

3 Minimizing the Maximum Subdomain Degree

There are two different approaches for computing a k-way partitioning of a hypergraph. One is based on recursive bisectioning and the other on direct k-way partitioning [18]. In recursive bisectioning, the overall partitioning is obtained by initially bisecting the hypergraph to obtain a two-way partitioning. Then, each of these parts is further bisected to obtain a four-way partitioning, and so on. Assuming that k is a power of two, then the final k-way partitioning can be obtained in $\log(k)$ such steps (or after performing $k-1$ bisections). In this approach, each partitioning step usually takes into account information from only two partitions, and as such it does not have sufficient information to explicitly minimize the maximum subdomain degree of the resulting k-way partitioning. In principle, additional information can be propagated down at each bisection level to account for the degrees of the various subdomains. For example, during each bisection step, the change in the degrees of the adjacent subdomains can be taken into account

(either explicitly or via variations of terminal-propagation-based techniques [16]) to favor solutions that in addition to minimizing the cut also reduce these subdomain degrees. However, the limitation of such approaches is that they end-up *over-constraining* the problem because not only they try to reduce the maximum subdomain degree of the final k-way partitioning, but they also try to reduce the maximum degree of the intermediate lower-k partitioning solutions.

For this reason, approaches based on direct k-way partitioning are better suited for the problem of minimizing the maximum subdomain degree, as they provide a *concurrent* view of the entire k-way partitioning solution. The ability of direct k-way partitioning to optimize objective functions that depend on knowing how the hyperedges are partitioned across all k partitions has been recognized by various researchers, and a number of different algorithms have been developed to minimize objective functions such as the sum-of-external-degrees, scaled cost, absorption *etc*. [30, 5, 10, 25, 34]). Moreover, direct k-way partitioning can potentially produce much better solutions than a method that computes a k-way partitioning via recursive bisection. In fact, in the context of a certain classes of graphs it was shown that recursive bisectioning can be up to an $O(\log n)$ factor worse than the optimal solution [33].

However, despite the inherent advantage of direct k-way partitioning to naturally model much more complex objectives, and the theoretical results which suggest that it can lead to superior partitioning solutions, a number of studies have shown that existing direct k-way partitioning algorithms for hypergraphs, produce solutions that are in general inferior to those produced via recursive bisectioning [30, 10, 25, 34]. The primary reason for that is the fact that computationally efficient k-way partitioning refinement algorithms are often trapped into local minima, and usually require much more sophisticated and expensive optimizers to climb out of them.

To overcome these conflicting requirements and characteristics, our algorithms for minimizing the maximum subdomain degree combine the best features of the recursive bisectioning and direct k-way partitioning approaches. We achieve this by treating the minimization of the maximum subdomain degree as a post-processing problem to be performed once a high-quality k-way partitioning has been obtained. Specifically, we use existing state-of-the-art multilevel-based techniques [19, 22] to obtain an initial k-way solution via repeated bisectioning, and then refine this solution using various k-way partitioning refinement algorithms that (i) explicitly minimize the maximum subdomain degree, (ii) ensure that the cut does not significantly increase, and (iii) ensure that the balancing constraints of the resulting k-way partitioning are satisfied.

This approach has a number of inherent advantages. First, by building upon a cut-based k-way partitioning, it leverages the huge body of existing research on this topic, and it can benefit from future improvements. Second, in terms of cut, its initial k-way solution is of extremely high-quality; thus, allowing us to primarily focus on minimizing the maximum subdomain degree without being overly concerned about the cut of the final solution (as long as the partitioning is not significantly perturbed). Third, it allows for a user-adjustable and predictable framework in which the user can specify how much (if any) deterioration on the cut he or she is willing to tolerate in order to reduce the maximum subdomain degree.

To actually perform the maximum subdomain-degree focused k-way refinement we developed two classes of algorithms. Both of them treat the problem as a multi-objective optimization problem but they differ on the starting point of that refinement. The first algorithm called "Direct Multi-Phase Refinement" directly optimizes the multi-objective cost using k-way V-cycle framework [19], while the second algorithm called "Aggressive Multi-Phase Refinement" utilizes refinement strategies that enable large scale perturbations of the solution space. Details on the exact multi-objective formulation is provided in the rest of this section and the two refinement algorithms are described in subsequent sections.

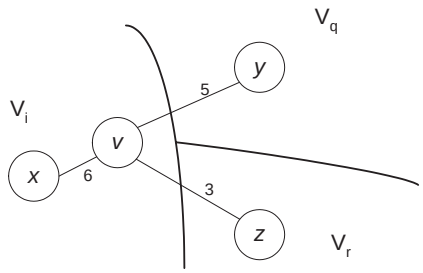

Figure 3: An example in which the objectives of maximum subdomain degree and cut are in conflict with each other. Lets say V_i is the subdomain with the maximum degree. If v is moved to either V_q or V_r it will increase the cut either by one or three, respectively. However both moves will reduce the maximum subdomain degree by two (5+3-6).

3.1 Multi-Objective Problem Formulation

In general, the objectives of producing a k-way partitioning that both minimize the cut and the maximum subdomain degree are reasonably well correlated with each other, as the partitionings with low cuts will also tend to have low maximum subdomain degrees. However, this correlation is not perfect, and these two objectives can actually be at odds with each other. That is, a reduction in the maximum subdomain degree may only be achieved if the cut of the partitioning is increased. This situation arises with vertices that are adjacent to vertices that belong to more than two subdomains. For example, consider a vertex v that belongs to the maximum degree partition V_i and let V_q and V_r be two other partitions such that v is connected to vertices in V_i, V_q, and V_r. Now, if $\text{ED}_q(v) - \text{ID}_i(v) < 0$ and $\text{ED}_r(v) - \text{ID}_i(v) < 0$, then the move of v to either partitions V_q or V_r will increase the cut but if $\text{ED}_q(v) + \text{ED}_r(v) - \text{ID}_i(v) > 0$, then moving v to either V_q or V_r will actually decrease V_i's subdomain degree. One such scenario is illustrated in Figure 3, in which vertex v from partition V_i is connected to vertices x, y, and z of partitions V_i, V_q and V_r, respectively, and the weights of the respective edges are 6, 5, and 3. Moving vertex v from partition V_i to either partitions V_q or V_r will reduce the subdomain degree of V_i; however, either of these moves will increase the overall cut of the partitioning. For example, if v moves to V_q, the subdomain degree of V_i will reduce from 8 to 6, whereas the overall cut will increase from 8 to 9. This discussion suggests that in order to develop effective algorithms that explicitly minimize the maximum subdomain degree and the cut, these two objectives need to be coupled together into a multi-objective framework that allows the optimization algorithm to intelligently select the preferred solution.

The problem of multi-objective optimization within the context of graph and hypergraph partitioning has been extensively studied in the literature [31, 1, 36, 29, 27] and two general approaches have been developed for combining multiple objectives. The first approach keeps the different objectives separate and couples them by assigning to them different priorities. Essentially in this scheme, a solution that optimizes the highest priority objective the most is always preferred and the lower priority objectives are used as *tie-breakers* (*i.e.*, used to select among equivalent solutions in terms of the higher priority objectives). The second approach creates an explicit multi-objective function that numerically combines the individual functions. For example, a multi-objective function can be obtained as the weighted sum of the individual objective functions. In this scheme, the choice of the weight values is used to determine the relative importance of the various objectives. One of the advantages of such an approach is that it tends to produce somewhat more natural and predictable solutions as it will prefer solutions that to a certain extent, optimize all different objective functions.

In our algorithms we used both of these methods to combine the two different objectives. Specifically, our priority-

based scheme produces a multi-objective solution in which the maximum subdomain degree is the highest priority objective and the cut is the second highest. This choice of priorities was motivated by the fact that within our framework, the solution is already at a local minima in terms of cut; thus, focusing on the maximum subdomain degree is a natural choice.

Our combining multi-objective function couples the different objectives using the formula

$$\text{Cost} = \alpha(\text{MaximumDegree}) + \text{Cut}, \tag{1}$$

where *MaximumDegree* is the maximum subdomain degree, *Cut* is the hyperedge cut, and α is a user-specified weight indicating the importance of maximum subdomain degree relative to the cut. Selecting the proper value of this parameter is, in general, problem dependent. As discussed earlier, in many cases the maximum subdomain degree can be only reduced by increasing the overall cut of the partitioning. As a result, in order for Equation 1 to provide meaningful maximum subdomain degree reduction, α should be greater than 1.0. Moreover, since the cut worsening moves that lead to improvements in the maximum subdomain degree are those in which the moved vertices are connected to vertices of different partitions (*i.e.*, corner vertices), then the value α should be an increasing function on the number of partitions k; thus, allowing for the movement of vertices that are adjacent to many subdomains (as long as such moves reduce the maximum subdomain degree). The sensitivity on these parameters is further studied in the experiments shown in Section 6.

In addition, in both of these schemes, we break ties in favor of solutions that lead to lower sum-of-external-degrees. This was motivated by the fact that lower SOED solutions may lead to subsequent improvements in either one of the main objective functions. Also, if a gain of the move is tied even after considering SOED, the ability of the move to improve area balancing is considered for tie breaking.

4 Direct Multi-Phase Refinement

Our first k-way refinement algorithm for the multi-objective problem formulations described in Section 3.1 is based on the *multi-phase refinement* approach implemented by hMETiS and was initially described in [19]. The idea behind multi-phase refinement is quite simple. It consists of two phases, namely a coarsening and an uncoarsening phase. The uncoarsening phase is identical to the uncoarsening phase of the multilevel hypergraph partitioning algorithm [19]. The coarsening phase, called *restricted coarsening* [19], however is somewhat different, as it preserves the initial partitioning that is input to the algorithm. Given a hypergraph G and a partitioning P, during the coarsening phase a sequence of successively coarser hypergraphs and their partitionings is constructed. Let (G_i, P_i) for $i = 1, 2, \ldots, m$, be the sequence of hypergraphs and partitionings. Given a hypergraph G_i and its partitioning P_i, restricted coarsening will collapse vertices together that belong to only one of the two partitions. The partitioning P_{i+1} of the next level coarser hypergraph G_{i+1} is computed by simply inheriting the partition from G_i. By constructing G_{i+1} and P_{i+1} in this way we ensure that the number of hyperedges cut by the partitioning is identical to the number of hyperedges cut by P_i in G_i.

The set of vertices to be collapsed together in this restricted coarsening scheme can be selected by using any of the coarsening schemes that have been previously developed [19]. In our algorithm, we use the first-choice (FC) scheme [25] as our default, since it leads to the best overall solutions [22]. The FC scheme is derived by modifying the commonly used edge-coarsening scheme. In the edge-coarsening scheme, a vertex is randomly selected and

it is merged with a highly connected and unmatched neighbor. The connectivity to the neighbors is estimated by representing each hyperedge by a clique of edges each with the weight of $w(e)/(|e|-1)$ and by summing the weights of edges common to each neighbor and the vertex in consideration. The FC coarsening scheme is derived from the edge-coarsening scheme by relaxing the requirement that a vertex is matched only with another unmatched vertex. Specifically, in the FC coarsening scheme, the vertices are again visited in a random order. However, for each vertex v, all vertices (both matched and unmatched) that belong to hyperedges incident to v are considered, and the one that is connected via the edge with the largest weight is matched with v, breaking ties in favor of unmatched vertices. The FC scheme tends to remove a large amount of the exposed hyperedge-weight in successive coarse hypergraphs, and thus, makes it easy to find high-quality initial partitionings that require little refinement during the uncoarsening phase.

Due to the randomization in the coarsening phase, successive runs of the multi-phase refinement algorithm can lead to additional improvements of the partitioning solution. For this reason, in our algorithm we perform multiple such iterations and the entire process is stopped when the solution quality does not improve in successive iterations. Such an approach is identical to the V-cycle refinement algorithm used by hMETiS [22].

The actual k-way partitioning refinement at a given level during the uncoarsening phase is performed using a greedy algorithm that is motivated by a similar algorithm used in the direct k-way partitioning algorithm of hMETiS. More precisely, the greedy k-way refinement algorithm works as follows. Consider a hypergraph $G = (V, E)$, and its partitioning vector P. The vertices are visited in a random order. Let v be such a vertex, let $P[v] = a$ be the partition that v belongs to. If v is a node internal to partition a then v is not moved. If v is at the boundary of the partition, then v can potentially be moved to one of the partitions $N(v)$ that vertices adjacent to v belong to (the set $N(v)$ is often referred to as the *neighborhood* of v). Let $N'(v)$ be the subset of $N(v)$ that contains all partitions b such that movement of vertex v to partition b does not violate the balancing constraint. Now the partition $b \in N'(v)$ that leads to the greatest positive reduction in the multi-objective function is selected and v is moved to that partition.

5 Aggressive Multi-Phase Refinement

One of the potential problems with the multi-objective refinement algorithm described in Section 4 is that it is limited to the extent in which it can make large-scale perturbations on the initial k-way partitioning produced by the cut-focused recursive-bisectioning algorithm. This is due to the combination of two factors. First, the greedy, non-hill climbing nature of its refinement algorithm limits the perturbations that are explored, and second, since it is based on an FM-derived framework, it is constrained to make moves that do not violate the balancing constraints of the resulting solution. As a result (shown later in our experiments (Section 6)), it tends to produce solutions that retain the low-cut characteristics of the initial k-way solution, but it does not significantly reduce the maximum subdomain degree. Ideally, we will like a multi-objective refinement algorithm that is capable of effectively *exploring* the entire space of possible solutions in order to select the one that best optimizes the particular multi-objective function.

Toward this goal, we developed two multi-objective refinement algorithms that allows large-scale perturbations of the partitioning produced by the recursive bisectioning algorithm. These algorithms are described in detail in the following sections.

5.1 Bottom-Up Aggressive Multi-phase Refinement

The first algorithm, referred to as *bottom-up aggressive multi-phase refinement*, consists of five major steps (outlined in Figure 4) and operates as follows.

In the first step, given the initial k-way partitioning, the algorithm proceeds to further subdivide each of these partitions into 2^l parts (where l is a user specified parameter). These subdivisions are performed using a min-cut hypergraph partitioning algorithm, resulting in a high-quality fine-grain partitioning. During the second step, this $2^l k$-way partitioning is refined using the direct multi-phase refinement algorithm described in Section 4 to optimize the particular multi-objective function. Each of the resulting $2^l k$ partitions are then collapsed into single nodes, that we will refer to them as *macro nodes*.

During the third step, a k-way partitioning of these macro nodes is computed such that each partition has exactly 2^l macro nodes. The goal of this macro-node partitioning is to obtain an initial partitioning that has low maximum subdomain degree and is achieved by greedily combining macro nodes that lead to the smallest maximum subdomain degree as follows. For each pair of macro-nodes u_i and u_j ($i < j$), let $v_{i,j}$ be the node obtained by merging u_i and u_j, and let deg(u_i) and deg($v_{i,j}$) be the degrees of macro-node u_i and merged node $v_{i,j}$, respectively. For $l = 1$, the algorithm orders all possible macro-node pairs in non-increasing order based on the maximum degree of its constituent macro-nodes (i.e., for each pair $v_{i,j}$ it considers $\max\{\text{deg}(u_i), \text{deg}(u_j)\}$) and the macro-node pairs that have the same maximum degree are ordered in non-decreasing order of their resulting degree (i.e., for each pair $v_{i,j}$ it considers deg($v_{i,j}$)). The algorithm then traverses the list in that order to identify the pairs of *unmatched* macro-nodes to form the initial partitioning. As a result of this traversal order, the algorithm provides the highest flexibility to the macro-nodes that have high degree and tries to combine them with the macro-nodes that will lead to pairs that have the smallest degree. However, for $l > 1$, since a direct extension of such an approach is not computationally feasible (the number of combinations that needs to be considered increases exponentially with l), the algorithm obtains the partitioning in a bottom-up fashion by repeatedly applying the above scheme l times. In addition, after each round of pairings, the macro-node-level partitioning is further refined by applying a pair-wise macro-node swapping algorithm described in Section 5.1.1.

In the fourth step, the quality in terms of the particular multi-objective function of the resulting macro-node level partitioning is improved using a pair-wise macro-node swapping algorithm (described in Section 5.1.1). This algorithm operates at the macro-node level and selects two macro-nodes, each one from a different partition, and swaps the partitions that they belong so that to improve the overall quality of the solution. Since by construction, each macro node is approximately of the same size, such swaps almost always lead to feasible solutions in terms of the balance constraint. The use of such a refinement algorithm was the primary motivation behind the development of the aggressive multi-phase algorithm as it allows us to move large portions of the hypergraph between partitions without having to either violate the balancing constraints or rely on a sequence of small vertex-moves in order to achieve the same effect. Moreover, because by construction, each macro-node corresponds to a good cluster (as opposed to a random collection of nodes) such swaps can indeed lead to improved quality very efficiently.

Finally, in the fifth step, the macro-node based partitioning is used to induce a partitioning of the original hypergraph, which is then further improved using the direct multi-phase refinement algorithm described in Section 4.

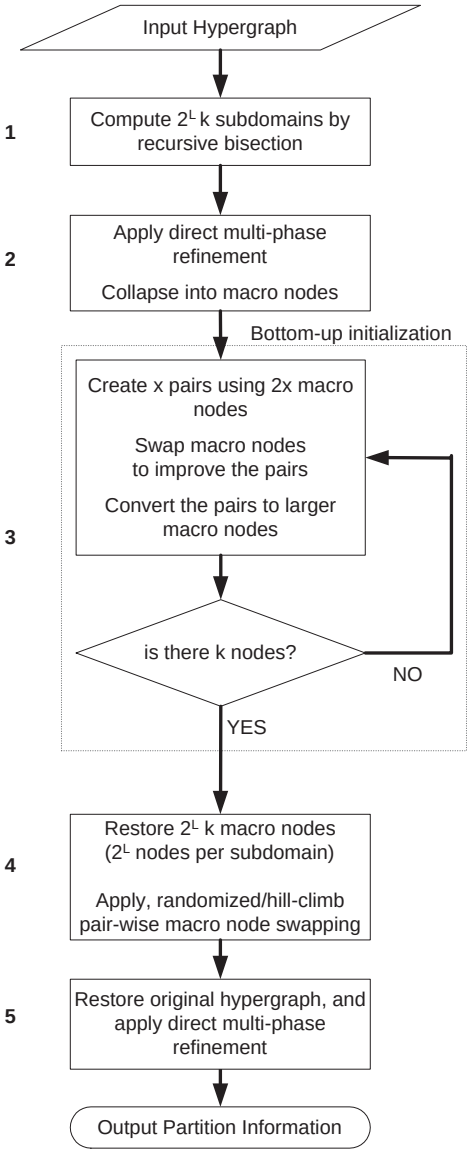

Figure 4: The various steps of the bottom-up aggressive multi-phase refinement algorithm.

5.1.1 Macro-node Partitioning Refinement

We developed two algorithms for refining a partitioning solution at the macro-node level. The differences between the two algorithms are the method used to identify the pairs of macro nodes to be swapped and the policy used in determining whether or not a particular swap will be accepted. Details on these two schemes are provided in the next two sections.

Randomized Pair-wise Node Swapping In this scheme, two nodes belonging to different partitions are randomly selected and the quality of the partitioning resulting by their swap is evaluated in terms of the particular multi-objective function. If that swap leads to a better solution, the swap is performed, otherwise it is not. Swaps that do not improve or degrade the particular multi-objective function are also allowed, as they often introduce desirable perturbations. The primary motivation for this algorithm is its low computational complexity, and in practice it produces very good results. Also, when there are two nodes per subdomain, the randomized pair-wise node swapping can be done quite efficiently by pre-computing the cut and degree of all possible pairings and storing them in a 2D table. This loop-up based swapping takes less than one second to evaluate the cost of one million swaps on a 1.5 GHz workstation.

Coordinated Sequence of Pair-wise Node Swaps One of the limitations of the previous scheme is that it lacks the ability to climb out of local minima as it does not allow any swaps that decrease the value of the objective function. To overcome this problem, we developed a heuristic refinement algorithm that can be considered an extension of the classical Kernighan-Lin algorithm [26] for k-way refinement that operates as follows.

The algorithm consists of a number of iterations. During each iteration it identifies and performs a sequence of macro-nodes swaps that improve the value of the objective function and terminates when no such sequence can be identified within a particular iteration. Each of these iterations is performed as follows. Let k be the number of partitions, m the number of macro-nodes, and $q = m/k$ the number of macro-nodes per partition. Since each macro-node v in partition V_i can be swapped with any macro-node belonging to a different partition, there are a total of $m(m-q)$ possible pairs of macro-nodes that can be swapped. For each of these swaps, the algorithm computes the improvement in the value of the objective function (i.e., the *gain*) achieved by performing it, and inserts all the $m(m-q)$ possible swaps into a max-priority queue based on this gain value. Then it proceeds to repeatedly (i) extract from this queue the macro-node pair whose swap leads to the highest gain, (ii) modify the partitioning by performing the macro-node swap, (iii) record the current value of the objective function, and (iv) update the gains of the macro-node pairs in the priority queue to reflect the new partitioning. Once the priority queue becomes empty, the algorithm determines the point within this sequence of swaps that resulted in the best value of the objective function and reverts the swaps that it performed after that point. An outline of the single iteration of this "hill-climb swap" algorithm is presented in Algorithm 1.

Due to the global nature of the maximum subdomain degree cost, if a macro-node swap changes the value of the maximum subdomain degree, the gains of all the pairwise swaps that are still in the priority queue needs to be recomputed. However, if a swap does not change the value of the maximum subdomain degree, then only the gains of the macro-node pairs that contain nodes adjacent to those being swapped need to be recomputed. Since only a small fraction of the swaps will end up changing the value of the maximum subdomain degree, the cost of updating the priority queue is relatively small. Despite this, since the algorithm needs to evaluate all $m(m-q)$ possible pairs of swaps, its runtime complexity is significantly higher than that of the randomized swapping algorithm.

It is easy to see that this algorithm is quite similar in spirit to the Kernighan-Lin algorithm with the key difference

Algorithm 1 Hill-climbing algorithm for identifying a sequence of pair-wise macro node swaps to reach a lower cost

```
Compute initial gain values for all possible pairs
Insert them in a priority queue

while Pairs exist in priority queue do
   Pop the highest gain pair
   Make the swap
   "Lock" the pair

   if Cost is minimum then
      Record roll back point
      Record new minimum cost
   end if

   if Maximum subdomain degree changed then
      Update the gain values of all pairs remaining in priority queue.
   else
      Update the gain values of affected pairs remaining in priority queue.
   end if
end while

Roll back to minimum cost point (i.eundo all swaps after the minimum cost point in reverse order)
```

being that the priority queue stores the effect of a pairwise macro-node swap as opposed to the effect of a single vertex move. This swapping-based view allows this algorithm to operate for an arbitrary number of partitions k.

5.2 Top-Down Aggressive Multi-Phase Refinement

The key parameter of the aggressive refinement scheme described in Section 5.1 is the value of l, which controls the granularity of the macro-nodes that are used. The effectiveness of the overall refinement approach can be affected both for small as well as large values of l. Small values may lead to large macro-nodes whose swaps do not improve the quality, whereas large values may lead to small macro-nodes that require a *coordinated* sequence of swaps to achieve the desired perturbations. Moreover, large values of l have the additional drawback of increasing the overall runtime of the algorithm as it requires more time to obtain the initial clusters and more refinement time.

In developing the bottom-up aggressive multi-phase refinement algorithm we initially expected that its performance will continue to improve as we increase the value of l, until the point at which the size of the macro-nodes will become so small so that this macro-node based partitioning does not provide any advantages over the unclustered hypergraph. Our experimental results (presented later in Section 6) partially verified this intuition but also showed that the point after which we obtain no improvements is actually much higher from what we had expected. In particular, our results will show that as l increases from zero to two, the bottom-up scheme can achieve progressively better results, but its performance for $l \geq 3$ is actually worse than that for $l = 2$. In analyzing this behavior we realized that there is another parameter that affects the effectiveness of this scheme and has to do with the bottom-up nature of the k-way partitioning that is computed in the third step of the algorithm.

Recall from Section 5.1 that for $l > 1$ the k-way macro-node partitioning is computed by repeatedly merging successively larger partitions, followed by a swapping-based refinement. For example, when $l = 3$, we first merge pairs of macro-nodes to obtain a $4k$-way partitioning, then merge these partitions to obtain a $2k$-way partitioning,

and finally obtain the desired k-way partitioning by merging these $2k$ partitions. Moreover, between these merging operations we apply the swap-based refinement to optimize the overall quality of the $4k$-, $2k$-, and k-way partitioning. The problem with this approach arises from the fact that as we move from the $8k$ macro-nodes to a $4k$-, $2k$-, and k-way partitioning we optimize the intermediate solutions so that they minimize their respective maximum subdomain degree. However, the maximum subdomain degree at the $4k$-way (or $2k$-way) partitioning level may not directly affect the maximum subdomain degree at the desired k-way partitioning level. In fact, due to the heuristic nature of the refinement strategies, the fact that these intermediate maximum subdomain degrees have been optimized may reduce our ability to obtain low maximum subdomain degrees at the k-way level. Thus, an inherent problem with the bottom-up scheme is the fact that it ends up spending a lot of time refining intermediate solutions that may not directly or indirectly benefit our ultimate objective, which is to obtain a k-way partitioning that minimizes the maximum subdomain degree.

Having taken cognizance of this phenomenon, we devised another aggressive multi-phase refinement algorithm that employs a top-down framework, which allows it to always optimize the objective function within the context of the k-way partitioning solution. The overall structure of the algorithm is shown in Figure 5 and shares many of the characteristics of the bottom-up algorithm. In particular, the last two steps of these algorithms are identical and differ only in the first three steps, out of which the third step represents the key conceptual difference.

The top-down algorithm starts by computing a k-way partitioning that minimizes the cut using recursive bisectioning and it further refines the solution using the direct multi-phase refinement algorithm of Section 4 that takes into consideration the multi-objective nature of the problem. During the third step it performs l levels of *aggressive* refinement by repeatedly creating macro-nodes and swapping them between the k partitions. Specifically, during the ith refinement level, it splits each one of the k partitions into 2^i sub-partitions (resulting in a total of $2^i k$ sub-partitions), creates a macro node for each sub-partition, and optimizes the overall k-way solution by applying a swap-based macro-node refinement algorithm (Section 5.1.1). Note that the initial macro-node-level k-way partitioning is obtained by inheriting the current k-way partitioning of the hypergraph. Since this approach always focuses on optimizing the solution at the k-way partitioning level, it does not suffer from the limitations of the bottom-up scheme. Moreover, as l increases, this scheme considers for swapping successively smaller macro-nodes, which allows it to perform large-scale perturbations at multiple scales (as it was the case with the bottom-up scheme).

6 Experimental Results

We experimentally evaluated our multi-objective partitioning algorithms on the 18 hypergraphs that are part of the ISPD98 circuit partitioning benchmark suite [3] (with unit area). The characteristics of these hypergraphs are shown in Table 2. For each of these circuits, we computed a 4-, 8-, 16-, 32-, and 64-way partitioning solution using the recursive bisection-based partitioning routine of hMETIS 1.5.3 [22] and the various algorithms that we developed for minimizing the maximum subdomain degree. The hMETIS solutions were obtained by using a 49–51 bisection balance constraint and hMETIS's default set of parameters. Since these balance constraints are specified at each bisection level, the final k-way partitioning may have a somewhat higher load imbalance. To ensure that the results produced by our algorithm can be easily compared against those produced by hMETIS, we used the resulting minimum and maximum partition sizes obtained by hMETIS as the balancing constraints for our multi-objective k-way refinement algorithm.

The quality of the solutions produced by our algorithm and those produced by hMETIS were evaluated by looking at three different quality measures, which are the maximum subdomain degree, the cut, and the average subdomain

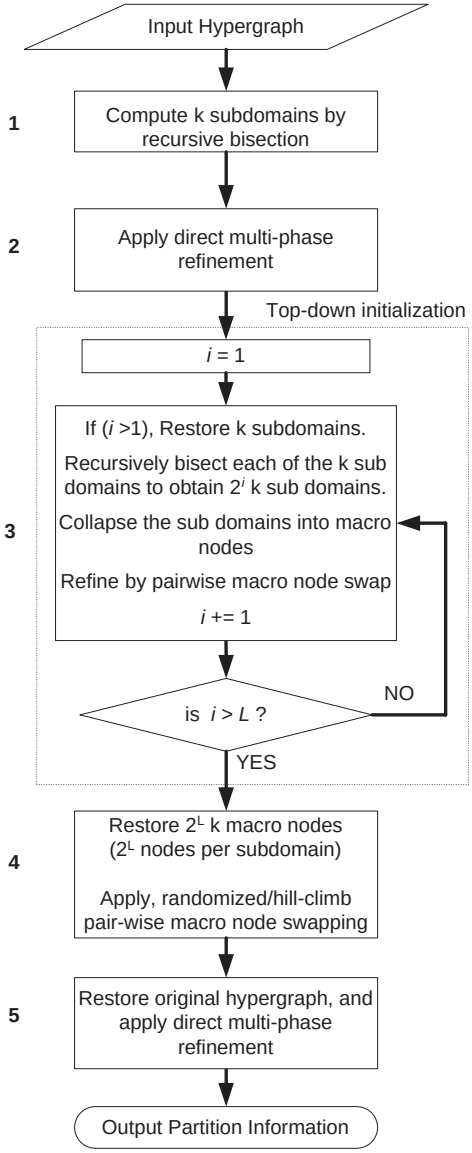

Figure 5: The various steps of the top-down aggressive multi-phase refinement algorithm.

Benchmark	No. of vertices	No. of hyperedges
ibm01	12506	14111
ibm02	19342	19584
ibm03	22853	27401
ibm04	27220	31970
ibm05	28146	28446
ibm06	32332	34826
ibm07	45639	48117
ibm08	51023	50513
ibm09	53110	60902
ibm10	68685	75196
ibm11	70152	81454
ibm12	70439	77240
ibm13	83709	99666
ibm14	147088	152772
ibm15	161187	186608
ibm16	182980	190048
ibm17	184752	189581
ibm18	210341	201920

Table 2: The characteristics of the hypergraphs used to evaluate our algorithm.

degree. To ensure the statistical significance of our experimental results, these measures were averaged over ten different runs for each particular set of experiments.

Furthermore, due to space constraints, our comparisons against hMETIS are presented in a summary form, which shows the relative maximum subdomain degree (*RMax*), relative cut (*RCut*), and relative average degree (*RDeg*) achieved by our algorithms over those achieved by hMETIS recursive bisection averaged over the entire set of 18 benchmarks. To ensure the meaningful averaging of these ratios, we first took their $\log_2$-values, calculated their mean μ, and then used 2^μ as their average. This geometric mean of ratios ensures that ratios corresponding to comparable degradations or improvements (*i.e.*, ratios that are less than or greater than one) are given equal importance.

6.1 Direct Multi-Phase Refinement

Our first set of experiments was focused on evaluating the effectiveness of the direct multi-phase refinement algorithm described in Section 4. Toward this goal we performed a series of experiments using both formulations of the multi-objective problem definition described in Section 3.1. The performance achieved in these experiments relative to those obtained by hMETIS's recursive bisectioning algorithm is shown in Table 3. Specifically, this table shows four sets of results. The first set uses the priority-based multi-objective formulation whereas the remaining three sets use Equation 1 to numerically combine the two different objectives. The objectives were combined using three different values of α, namely $1, 2$, and k (where k is the number of partitions that is computed).

The results of Table 3 show that irrespective of the number of partitions or the particular multi-objective formulation, the direct multi-phase refinement algorithm produces solutions whose average quality along each one of the three different quality measures is better than the corresponding solutions produced by hMETIS. As expected, the relative improvements are higher for the maximum subdomain degree. In particular, depending on the number of partitions, the direct multi-phase refinement algorithm reduces the maximum subdomain degree by 5% to 15%. The relative improvements increase as the number of partitions increase, because as the results in Table 1 showed, these are the

	Prioritized			Combined, $\alpha = 1$			Combined, $\alpha = 2$			Combined, $\alpha = k$		
k	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg
4	0.955	0.981	0.948	0.940	0.967	0.934	0.928	0.964	0.931	0.929	0.967	0.934
8	0.890	0.967	0.913	0.877	0.947	0.892	0.886	0.952	0.897	0.881	0.959	0.906
16	0.884	0.969	0.898	0.876	0.958	0.886	0.886	0.965	0.894	0.886	0.966	0.894
32	0.865	0.967	0.886	0.874	0.959	0.874	0.871	0.963	0.877	0.870	0.964	0.878
64	0.851	0.970	0.880	0.864	0.966	0.872	0.876	0.970	0.875	0.859	0.969	0.875

Table 3: Direct Multi-Phase Refinement Results. *RMax*, *RCut*, and *RDeg* are the average maximum subdomain degree, cut, and average subdomain degree, respectively of the multi-objective solution relative to hMETIS. Numbers less than one indicate that the multi-objective algorithm produces solutions that have lower maximum subdomain degree, cut, or average subdomain degree than those produced by hMETIS.

partitioning solutions in which the maximum subdomain degree is significantly higher than the average and thus there is significantly more room for improvement.

Furthermore, the direct multi-phase refinement algorithm also leads to partitionings that on the average have lower cut and average subdomain degree. Specifically, the cut tends to improve by 1% to 4%, whereas the average subdomain degree improves by 5% to 13%. Finally, comparing the different multi-objective formulations we can see that in general, there are very few differences between them, with both of them leading to comparable solutions.

6.2 Aggressive Multi-Phase Refinement

Our experimental evaluation of the aggressive multi-phase refinement schemes (described in Section 5) focused along two directions. First, we designed a set of experiments to evaluate the effectiveness of the macro-node-level partitioning refinement algorithms used by these schemes and second, we performed a series of experiments that were designed to evaluate the effectiveness of the bottom-up and top-down schemes within the context of aggressive refinement.

6.2.1 Evaluation of Macro-node Partitioning Refinement Algorithms

To directly evaluate the relative performance of the two refinement algorithms described in Section 5.1.1 we performed a series of experiments using a simple version of the aggressive refinement schemes. Specifically, we computed a $2^l k$-way partitioning, collapsed each partition into a macro node, and obtained an initial k-way partitioning of these macro nodes using a random assignment. This initial partitioning was then refined using the two macro-node partitioning refinement algorithms—randomized swap and hill-climbing swap. This experiment was performed for each one of the circuits in our benchmark suite and the overall performance achieved by the two algorithms for $k = 8, 16, 32$ and $l = 1, 2$ relative to those obtained by hMETIS's recursive bisectioning algorithm is shown in Table 4. Note that for this set of experiments, the two objectives of maximum subdomain degree and cut were combined using a priority scheme, which uses the minimization of the maximum subdomain degree as the primary objective.

From these results we can see that contrary to our initial expectations, the hill-climbing algorithm does not outperform the randomized randomized-swapping algorithm for all three performance metrics. Specifically, in terms of the cut (RCut) and the average degree (RDeg), the hill-climbing algorithm is superior to the randomized algorithm. For example, for $l = 2$ both of these measures are over 10% better than the corresponding measures for the randomized algorithm. However, in terms of the maximum subdomain degree (measured by RMax), the hill-climbing algorithm provides little advantage. In fact, its overall performance is slightly worse than the randomized scheme—leading to solutions whose maximum subdomain degree is about 1% to 3% higher for $l = 1$ and $l = 2$, respectively.

Randomized Swap								
	$l = 1$				$l = 2$			
k	RMax	RCut	RDeg	RTime	RMax	RCut	RDeg	RTime
8	0.953	1.001	1.005	1.979	0.999	1.113	1.137	2.201
16	0.930	1.018	1.027	1.931	0.917	1.127	1.174	2.321
32	0.905	1.017	1.036	1.883	0.845	1.112	1.168	2.323

Hill-climb Swap								
	$l = 1$				$l = 2$			
k	RMax	RCut	RDeg	RTime	RMax	RCut	RDeg	RTime
8	0.956	0.995	0.993	1.710	0.970	1.032	1.038	2.830
16	0.948	1.016	1.021	2.273	0.943	1.031	1.039	15.577
32	0.921	1.007	1.014	12.900	0.913	1.022	1.035	549.267

Table 4: Analysis of randomized vs hill-climb swapping.

The mixed performance of the hill-climbing algorithm and its inability to produce solutions that have lower maximum subdomain degree suggest that this type of refinement may not be well-suited for the step-nature of the maximum subdomain degree objective. Since there are relatively few macro-node swaps that affect the maximum subdomain degree, the priority queue used by the hill-climbing algorithm forces it to order the moves based on their gains with respect to the cut (as it is the secondary objective). Because of this, this refinement is very effective in minimizing RCut and RDeg but it does not affect RMax. In fact, as the results suggest, this emphasis on the cut may affect the ability of subsequent swaps to reduce the maximum subdomain degree. To see if this is indeed the case we performed another sequence of experiments in which we modified the randomized algorithm so that to perform the moves using the same priority-queue-based approach used by the hill-climbing scheme and terminated each inner-iteration as soon as the priority queue contained negative gain vertices (i.e., it did not perform any hill-climbing). Our experiments (not presented here) showed that this scheme produced results whose RMax was worse than that of the randomized and hill-climbing approaches but its RCut and RDeg were between those obtained by the randomized and the hill-climbing schemes—verifying our hypothesis that due to the somewhat conflicting nature of the two objectives, a greedy ordering scheme does not necessarily lead to better results.

The columns of Table 4 labeled "RTime" shows the amount of time required by the two refinement algorithms. As expected, the randomized algorithm is faster than the hill-climbing algorithm and its relative runtime advantage improves as the number of macro-nodes increases. Due to the mixed performance of the hill-climbing algorithm and its considerably higher computational requirements for large values of l, our subsequent experiments used only the randomized refinement algorithm.

6.2.2 Evaluation of Bottom-up and Top-down Schemes

Table 5 shows the performance achieved by the bottom-up and top-down aggressive multi-phase refinement schemes for $l = 1, \ldots, 3$, and $k = 4, 8, \ldots, 64$ relative to those obtained by hMeTiS's recursive bisectioning algorithm. Specifically, for each value of l and k, this table shows four sets of results. The first two sets (one for the bottom-up and one for the top-down scheme) were obtained using the priority-based multi-objective formulation whereas the remaining two sets used the combining scheme. Due to space constraints, we only present results in which the two objectives were combined using $\alpha = k$.

From these results, we can observe a number of general trends about the performance of the aggressive multi-phase refinement schemes and their sensitivities to the various parameters. In particular, as l increases from one to two

$l = 1$												
	Prioritized						Combined, $\alpha = k$					
	Bottom-Up			Top-Down			Bottom-Up			Top-Down		
k	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg
4	0.927	0.990	0.958	0.911	0.956	0.929	0.904	0.972	0.941	0.897	0.957	0.927
8	0.838	0.995	0.945	0.849	0.960	0.918	0.834	0.992	0.943	0.830	0.968	0.921
16	0.787	1.005	0.942	0.811	0.980	0.928	0.795	1.000	0.935	0.812	0.991	0.934
32	0.754	0.993	0.923	0.762	0.984	0.913	0.758	0.991	0.917	0.795	0.989	0.921
64	0.724	0.996	0.916	0.738	0.988	0.910	0.721	0.993	0.905	0.749	0.992	0.911

$l = 2$												
	Prioritized						Combined, $\alpha = k$					
	Bottom-Up			Top-Down			Bottom-Up			Top-Down		
k	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg
4	0.938	1.021	0.991	0.901	0.943	0.917	0.905	0.992	0.963	0.883	0.956	0.924
8	0.825	1.046	1.004	0.822	0.964	0.922	0.814	1.041	1.001	0.806	0.974	0.926
16	0.749	1.049	1.008	0.761	0.997	0.943	0.751	1.048	1.003	0.761	1.013	0.952
32	0.693	1.041	0.991	0.704	1.000	0.936	0.689	1.033	0.976	0.728	1.017	0.945
64	0.654	1.040	0.983	0.664	1.007	0.934	0.652	1.041	0.974	0.704	1.018	0.937

$l = 3$												
	Prioritized						Combined, $\alpha = k$					
	Bottom-Up			Top-Down			Bottom-Up			Top-Down		
k	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg	RMax	RCut	RDeg
4	1.007	1.121	1.091	0.899	0.953	0.927	0.950	1.058	1.029	0.877	0.950	0.919
8	0.848	1.119	1.088	0.815	0.974	0.931	0.842	1.109	1.073	0.796	0.982	0.934
16	0.759	1.101	1.070	0.748	1.000	0.945	0.754	1.077	1.034	0.759	1.022	0.966
32	0.697	1.095	1.059	0.682	1.006	0.943	0.700	1.064	1.010	0.722	1.023	0.954
64	0.701	1.100	1.052	0.645	1.012	0.941	0.663	1.066	1.006	0.683	1.024	0.945

Table 5: Aggressive Multi-Phase Refinement Results. *RMax*, *RCut*, and *RDeg* are the average maximum subdomain degree, cut, and average subdomain degree, respectively of the multi-objective solution relative to hMETiS. Numbers less than one indicate that the multi-objective algorithm produces solutions that have lower maximum subdomain degree, cut, or average subdomain degree than those produced by hMETiS.

(*i.e.*, each partition is further subdivided into two or four parts), the effectiveness of the multi-objective partitioning algorithm to produce solutions that have lower maximum subdomain degree compared to the solutions obtained by hMETiS, improves. In general, for $l = 1$, the multi-objective algorithm reduces the maximum subdomain degree by 7% to 28%, whereas for $l = 2$, the corresponding improvements range from 6% to 35%. However, these improvements lead to solutions in which the cut and the average subdomain degree obtained for $l = 2$ are somewhat higher than those obtained for $l = 1$. For example, for $l = 1$, the multi-objective algorithm is capable of improving the cut over hMETiS by 0% to 4%, whereas for $l = 2$, the multi-objective algorithm leads to solutions whose cut is up to 5% worse than those obtained by hMETiS. Note that these observations are to a large extent independent of the particular multi-objective formulation or the method used to obtain the initial macro-node-level partitioning.

For the reasons discussed in Section 5.2, the trend of successive improvements in the maximum subdomain degree does not hold for the bottom-up scheme any more for $l = 3$. In particular, the improvements in the maximum subdomain degree relative to hMETiS are in the range of 0%–35%, which are somewhat lower than the corresponding improvements for $l = 2$. On the other hand, the top-down scheme is able to further reduce the maximum subdomain degree when $l = 3$, leading to results that are 10% to 36% lower than the corresponding results of hMETiS. Note that this trend continues for higher values of l as well (due to space constraints these results are not reported here). These results suggest that the top-down scheme is better than the bottom-up scheme for large values of l. However, a closer

	Direct Scheme	Aggressive Schemes					
		Bottom-Up			Top-Down		
k		$l=1$	$l=2$	$l=3$	$l=1$	$l=2$	$l=3$
4	1.431	2.081	2.794	3.809	2.139	3.374	5.785
8	1.399	2.151	2.990	3.924	2.505	3.520	5.561
16	1.397	2.029	3.018	3.584	2.355	3.462	5.418
32	1.450	2.018	2.763	3.599	2.548	4.078	5.627
64	1.535	2.060	3.067	4.522	2.979	4.025	6.103

Table 6: The amount of time required by the multi-objective algorithms relative to that required by hMETIS.

inspection of the results reveals that for $l = 1$ and $l = 2$ and large values of k, the bottom-up scheme actually leads to solutions than are somewhat better than those obtained by the top-down scheme. We believe that this is due to the fact that for small values of l, the macro-node pairing scheme used by the bottom-up scheme to derive the macro-node level k-way partitioning (that takes into account all possible pairings of macro-nodes), is inherently more powerful than macro-node-level refinement used by the top-down scheme. This becomes more evident for large values of k, for which there is considerably more room for alternate pairings—resulting in relatively better results.

6.3 Comparison of Direct and Aggressive Multi-phase Refinement Schemes

Comparing the results obtained by the aggressive multi-phase refinement with the corresponding results obtained by the direct multi-phase refinement algorithm (Tables 5 and 3), we can see that in terms of the maximum subdomain degree, the aggressive scheme leads to substantially better solutions than those obtained by the direct scheme, whereas in terms of the cut and the average subdomain degree, the direct scheme is superior. These results are in agreement with the design principles behind these two multi-phase refinement schemes for the multi-objective optimization problem at hand, and illustrate that the former is capable of making relatively large perturbations on the initial partitioning obtained by recursive bisectioning, as long as these perturbations improve the multi-objective function. In general, the aggressive multi-phase refinement scheme with $l = 1$, dominates the direct scheme, as it leads to better improvements in terms of maximum subdomain degree and still improves over hMETIS in terms of cut and average degree. However, if the goal is to achieve the highest reduction in the maximum average degree, then the aggressive scheme with $l = 2$ should be the preferred choice, as it does so with relatively little degradation on the cut.

6.4 Runtime Complexity

Table 6 shows the amount of time required by the various multi-objective partitioning algorithms using either direct or aggressive multi-phase refinement. For each value of k and particular multi-objective algorithm, this table shows the total amount of time that was required to partition all 18 benchmarks relative to the amount of time required by hMETIS to compute the corresponding partitionings. From these results we can see that the multi-objective algorithm that uses the direct multi-phase refinement is the least computationally expensive and requires around 50% more time than hMETIS does. On the other hand, the time required by the aggressive multi-phase refinement schemes is somewhat higher and increases with the value of l. However, even for these schemes, their overall computational requirements are relatively small. For instance, in the case of the bottom-up scheme, for $l = 1$ and $l = 2$ (the cases in which it led to the best results) it only requires two and three times more time than hMETIS, respectively; and in the case of the top-down scheme its runtime is two to six times higher than that of hMETIS as l increases from one to three.

7 Conclusions

In this paper we described a family of multi-objective hypergraph partitioning algorithms for computing k-way partitionings that simultaneously minimize the cut and the maximum subdomain degree of the resulting partitions. Our experimental evaluation showed that these algorithms are quite effective in optimizing these two objectives with relatively low computational requirements. The key factor contributing to the success of these algorithms was the idea of focusing on the maximum subdomain degree objective once a good solution with respect to the cut has been identified. We believe that such a framework can be applied to a number of other multi-objective problems involving objectives that are reasonably well-correlated with each other.

The multi-objective algorithms presented here can be improved further in a number of directions. In particular, our results showed that the aggressive multi-phase refinement approach, especially when deployed in a top-down fashion, is very promising in reducing the maximum subdomain degree. Within this framework, our experiments revealed that due to the step-nature of the maximum subdomain degree objective, the hill-climbing macro-node refinement algorithm is not significantly more effective than the randomized swapping algorithm in reducing the maximum subdomain degree. Developing computationally scalable refinement algorithms that can successfully climb out of local minima for this type of objectives is still an open research problem whose solution can lead to even better results both for the partitioning problem addressed in this paper as well as other objective functions that share similar characteristics. Also, our work so far was focused on producing multi-objective solutions, which satisfy the same balancing constraints as those resulting from the initial recursive bisectioning based solution. However, additional improvements can be obtained by relaxing the lower-bound constraint. Our preliminary results with such an approach appears promising.

References

[1] C. Ababei, N. Selvakkumaran, K. Bazargan, and G. Karypis. Multi-objectivecircuit partitioning for cutsize and path-based delay minimization. In *Proceedings of ICCAD*, 2002. Also available on WWW at URL http://www.cs.umn.edu/~karypis.

[2] C. Alpert and A. Kahng. A hybrid multilevel/genetic approach for circuit partitioning. In *Proceedings of the Fifth ACM/SIGDA Physical Design Workshop*, pages 100–105, 1996.

[3] C. J. Alpert. The ISPD98 circuit benchmark suite. In *Proc. of the Intl. Symposium of Physical Design*, pages 80–85, 1998.

[4] C. J. Alpert, J. H. Huang, and A. B. Kahng. Multilevel circuit partitioning. In *Proc. of the 34th ACM/IEEE Design Automation Conference*, 1997.

[5] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning. *Integration, the VLSI Journal*, 19(1-2):1–81, 1995.

[6] J. Babb, R. Tessier, and A. Agarwal. Virtual wires: Overcoming pin limitations in FPGA-based logic emulators. In *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 142–151. IEEE Computer Society Press, 1993.

[7] S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. In *Proceedings of the sixth SIAM conference on Parallel Processing for Scientific Computing*, pages 711–718, 1993.

[8] T. Bui and C. Jones. A heuristic for reducing fill in sparse matrix factorization. In *6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 445–452, 1993.

[9] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Improved algorithms for hypergraph bipartitioning. In *Asia and South Pacific Design Automation Conference*, pages 661–666, 2000.

[10] J. Cong and S. K. Lim. Multiway partitioning with pairwise movement. In *Proceedings of ICCAD*, pages 512–516, 1998.

[11] J. Cong and M. L. Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in vlsi design. In *Proc. ACM/IEEE Design Automation Conference*, pages 755–760, 1993.

[12] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *International Conference on Tools with Artificial Intelligence*, pages 558–567, Newport Beach, 1997. IEEE.

[13] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *In Proc. 19th IEEE Design Automation Conference*, pages 175–181, 1982.

[14] S. Hauck and G. Borriello. An evaluation of bipartitioning technique. In *Proc. Chapel Hill Conference on Advanced Research in VLSI*, 1995.

[15] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories, 1993.

[16] B. Hendrickson, R. Leland, and R. V. Driessche. Enhancing data locality by using terminal propagation. In *Proceedings of the 29th Hawaii International Conference on System Science*, 1996.

[17] B. Hu and M. Marek-sadowska. Congestion minimization during placement without estimation. In *Proceedings of ICCAD*, pages 737–745, Nov 2002.

[18] G. Karypis. Multilevel hypergraph partitioning. In J. Cong and J. Shinnerl, editors, *Multilevel Optimization Methods for VLSI*, chapter 6. Kluwer Academic Publishers, Boston, MA, 2002.

[19] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. *IEEE Transactions on VLSI Systems*, 20(1), 1999. A short version appears in the proceedings of DAC 1997.

[20] G. Karypis, E. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[21] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *Proceedings of Supercomputing*, 1995. Also available on WWW at URL http://www.cs.umn.edu/~karypis.

[22] G. Karypis and V. Kumar. hMETIS 1.5: A hypergraph partitioning package. Technical report, Department of Computer Science, University of Minnesota, 1998. Available on the WWW at URL *http://www.cs.umn.edu/~metis*.

[23] G. Karypis and V. Kumar. METIS 4.0: Unstructured graph partitioning and sparse matrix ordering system. Technical report, Department of Computer Science, University of Minnesota, 1998. Available on the WWW at URL *http://www.cs.umn.edu/~metis*.

[24] G. Karypis and V. Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1), 1999. Also available on WWW at URL http://www.cs.umn.edu/~karypis. A short version appears in Intl. Conf. on Parallel Processing 1995.

[25] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. *VLSI Design*, 2000.

[26] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[27] P.Fishburn. *Decision and Value Theory*. J.Wiley & Sons, New York, 1964.

[28] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11(3):430–452, 1990.

[29] R.Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. J.Wiley & Sons, New York, 1976.

[30] L. Sanchis. Multiple-way network partitioning. *IEEE Trans. On Computers*, 38(1):62–81, 1989.

[31] K. Schloegel, G. Karypis, and V. Kumar. A new algorithm for multi-objective graph partitioning. In *Proceedings of EuroPar '99*, pages 322–331, 1999.

[32] S. Shekhar and D. R. Liu. Partitioning similarity graphs: A framework for declustering problmes. *Information Systems Journal*, 21(4), 1996.

[33] H. D. Simon and S.-H. Teng. How good is recursive bisection? Technical Report RNR-93-012, NAS Systems Division, NASA, Moffet Field, CA, 1993.

[34] M. Wang, S. K. Lim, J. Cong, and M. Sarrafzadeh. Multi-way partitioning using bi-partition heuristics. In *Proceedings of ASPDAC*, pages 441–446. IEEE, January 2000.

[35] S. Wichlund and E. J. Aas. On Multilevel Circuit Partitioning. In *Intl. Conference on Computer Aided Design*, 1998.

[36] P. Yu. *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions*. Plenum Press, New York, 1985.

[37] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *CIKM*, 2001.

[38] K. Zhong and S. Dutt. Algorithms for simultaneous satisfaction of multiple constraints and objective optimizaion in a placement flow with application to congestion control. In *Proceedings of DAC*, pages 854–859, 2002.